**1. Choix de React pour le Front-End**

"Nous avons choisi **React.js** pour le front-end en raison de sa flexibilité et de sa capacité à construire des interfaces utilisateur réactives et dynamiques. Sa large adoption dans l'industrie et sa communauté active facilitent le développement et la maintenance. De plus, React permet de créer des composants réutilisables, ce qui améliore l'efficacité du développement."

**2. Choix de Spring Boot pour le Back-End**

"**Spring Boot** a été retenu pour le back-end en raison de sa robustesse et de sa capacité à simplifier le développement d'API REST sécurisées et scalables. Il est idéal pour des applications de grande envergure, comme HomeSkolar, avec des exigences de performances élevées. Spring Boot offre également une intégration fluide avec des outils comme Spring Data JPA, ce qui facilite la gestion des données et des transactions."

**3. Choix de MariaDB pour la Base de Données**

"Nous avons opté pour **MariaDB**, une base de données relationnelle open-source, compatible avec MySQL, pour sa simplicité, ses performances et sa fiabilité. MariaDB est idéale pour gérer les données critiques de notre plateforme, comme les utilisateurs, les rendez-vous et les tâches, tout en garantissant une scalabilité efficace. Son intégration avec Spring Boot est simple, ce qui accélère le développement et garantit une gestion optimale des relations entre les entités."

Une **API REST (Representational State Transfer)** est un ensemble de règles qui permet à des applications de communiquer entre elles via le protocole HTTP, en utilisant des méthodes standardisées comme **GET**, **POST**, **PUT**, et **DELETE**.

### ****Principes clés :****

* **Stateless (sans état) :** Chaque requête entre le client et le serveur contient toutes les informations nécessaires, sans dépendance de l'état précédent.
* **Ressources :** Les données sont organisées en ressources, accessibles via des URL (par exemple, /users pour accéder à la liste des utilisateurs).
* **Format de données :** Les données sont généralement échangées au format **JSON** ou **XML**.

En résumé, une API REST permet d'interagir avec un serveur à travers des requêtes HTTP pour récupérer, ajouter, modifier ou supprimer des données de manière simple et structurée.

4o mini

### ****1. Clarification des exigences****

**Question :** Comment avez-vous clarifié les exigences vagues ou ambiguës dans le projet, notamment celles qui ne sont pas suffisamment détaillées dans les notes du client ?

**Réponse suggérée :**  
"Pour clarifier les exigences, j'ai d'abord découpé les besoins du client en **User Stories** détaillées, ce qui permet de mieux comprendre chaque fonctionnalité dans son contexte. Ensuite, j'ai utilisé des **hypothèses éclairées** basées sur des projets similaires pour combler les zones d'incertitude. Lorsque des éléments restaient flous, j'ai demandé des précisions au client afin d'affiner les besoins, en utilisant une approche **itérative** pour valider progressivement chaque fonctionnalité."

### ****2. Choix des technologies****

**Question :** Pourquoi avez-vous choisi ***React.js*** pour le front-end, ***Spring Boot*** pour le back-end et ***MariaDB*** pour la base de données ?

**Réponse suggérée :**  
"J'ai choisi **React.js** pour sa capacité à créer des interfaces utilisateur **réactives** et modulaires, avec une **grande communauté** et une **documentation abondante**. Pour le **back-end**, **Spring Boot** a été sélectionné pour sa **simplicité**, sa **sécurité** et sa **scalabilité**, ce qui est essentiel pour une application évolutive comme HomeSkolar. Enfin, **MariaDB** a été retenue pour ses **performances élevées**, sa **compatibilité avec MySQL**, et son intégration facile avec **Spring Boot**, permettant une gestion efficace des données relationnelles."

### ****3. Estimation des User Stories****

**Question :** Comment avez-vous estimé le temps nécessaire pour le développement des User Stories ?

**Réponse suggérée :**  
"Les estimations ont été réalisées en découpant chaque fonctionnalité en **sous-tâches spécifiques**. Pour chaque sous-tâche, j'ai pris en compte la **complexité technique**, le **temps de développement**, et les **tests nécessaires**. Les **imprévus** ont également été pris en compte en ajoutant une marge de sécurité. De plus, l'expérience de l'équipe et la durée des tâches similaires dans des projets passés m'ont aidé à évaluer les délais de manière plus réaliste."

### ****4. Pourquoi Agile ?****

**Question :** Pourquoi avez-vous choisi une approche ***Agile*** pour ce projet ?

**Réponse suggérée :**  
"L'approche **Agile** est idéale pour un projet comme HomeSkolar car elle permet de s'adapter aux besoins changeants du client. En utilisant des **sprints** courts et itératifs, nous pouvons livrer des **fonctionnalités opérationnelles** à chaque étape et obtenir des retours rapidement, afin d'ajuster le développement. Cela permet également de gérer efficacement les priorités et de garantir que les fonctionnalités les plus importantes sont livrées en premier."

### ****5. Sécurité de l'application****

**Question :** Comment garantissez-vous la ***sécurité*** des données, en particulier pour la gestion des comptes utilisateurs (élèves et tuteurs) ?

**Réponse suggérée :**  
"Pour assurer la sécurité des données, nous avons intégré **Spring Security**, un module robuste pour la gestion de l'**authentification** et de l'**autorisation**. Les **mots de passe** sont stockés de manière **cryptée** avec un algorithme sécurisé. En outre, l'**authentification par JWT** (JSON Web Token) permet de gérer l'accès aux ressources de manière sécurisée. Les données sensibles, comme les informations personnelles des utilisateurs, sont protégées par des **protocoles HTTPS** et des pratiques de gestion des sessions strictes."

### ****6. Gestion des performances et scalabilité****

**Question :** Comment comptez-vous gérer les ***performances*** et la ***scalabilité*** de l'application, en particulier si le nombre d'utilisateurs augmente ?

**Réponse suggérée :**  
"Pour garantir les **performances** et la **scalabilité**, nous avons choisi des technologies capables de supporter une croissance importante. **Spring Boot** et **MariaDB** sont optimisés pour gérer de **grandes quantités de données** et des **requêtes complexes** avec une faible latence. Nous utiliserons également des techniques de **mise en cache** et des **requêtes optimisées**. En cas de besoin, nous pourrons également envisager des **architectures microservices** pour découper l'application en services indépendants et faciliter l'ajout de nouvelles fonctionnalités sans impacter l'ensemble."

### ****7. Tests utilisateurs****

**Question :** Comment allez-vous ***tester*** l’application, notamment au niveau de l’expérience utilisateur ?

**Réponse suggérée :**  
"Nous commencerons par des **tests internes** pour vérifier la fonctionnalité des différentes parties de l'application. Ensuite, nous impliquerons un groupe de **tests utilisateurs** pour évaluer l'expérience réelle des élèves et des tuteurs. Ces tests permettront de récolter des **retours qualitatifs** pour améliorer l'interface et les fonctionnalités. De plus, nous effectuerons des **tests de charge** pour garantir que l'application peut supporter un grand nombre d’utilisateurs simultanés."

### ****8. Lancement et suivi****

**Question :** Quel est votre plan pour le ***lancement*** progressif de l’application et comment allez-vous gérer les ***retours des utilisateurs*** après le déploiement ?

**Réponse suggérée :**  
"Nous suivrons un **lancement progressif** en commençant par une **phase pilote** limitée à un groupe restreint d'élèves et de tuteurs. Cela nous permettra de tester l’application en conditions réelles avant de l’ouvrir à un plus grand nombre d’utilisateurs. Après chaque phase de lancement, nous collecterons des **retours utilisateurs** pour ajuster les fonctionnalités et corriger d'éventuels bugs. Le suivi sera effectué via un système de **ticketing** et un support utilisateur dédié."

### ****9. Gestion des erreurs et maintenance****

**Question :** Comment allez-vous gérer les erreurs et assurer la ***maintenance*** de l'application à long terme ?

**Réponse suggérée :**  
"Nous mettrons en place un système de **gestion des erreurs** centralisé, notamment avec **Spring Boot Actuator** pour surveiller l'état de l'application. Les erreurs seront loguées et envoyées à une équipe de support pour être traitées rapidement. Pour la **maintenance**, nous adopterons une approche **continue** avec des mises à jour régulières, des corrections de bugs et l'ajout de nouvelles fonctionnalités en fonction des besoins des utilisateurs. De plus, des tests unitaires et d’intégration garantiront que l’application reste stable après chaque modification."

### ****Conclusion****

Ces questions sont conçues pour tester votre compréhension du projet et des choix technologiques. Préparez des réponses détaillées et claires pour montrer que vous avez une vue d'ensemble de l'architecture et de la méthodologie utilisées. Si vous avez d'autres questions ou besoin de plus d'informations, n'hésitez pas à demander.